NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

USE OF WEBDAV TO SUPPORT A VIRTUAL FILE SYSTEM IN A COALITION ENVIRONMENT

by

Jeremiah A. Bradney

June 2006

Thesis Advisor:	Cynthia E. Irvine
Co-Advisor:	Thuy D. Nguyen

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Use of WebDAV to Support a Virtual File System in a Coalition Environment			**5. FUNDING NUMBERS**
6. AUTHOR(S) Jeremiah A. Bradney			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The Monterey Security Architecture (MYSEA) combines untrusted commercial-off-the-shelf components with specialized high-assurance trusted components to provide a trusted multilevel secure environment. However, MYSEA currently lacks a means of providing remote access to data on the MYSEA server, a vital service for users in any client-server environment. Access and interaction with both public and private server-resident data that is constrained by the underlying discretionary and mandatory access control policy enforcement mechanisms of the MYSEA server is required.

By enabling the use of WebDAV in MYSEA, this thesis provides a means for fulfilling the above requirement for secure remote access by creating a virtual web-based file system accessible from the MYSEA MLS network. This is accomplished by incorporating the mod_dav module into the Apache web server already installed on the MYSEA server. The use of a module required minimal changes to add the desired functionality. Integration of mod_dav is performed in three stages to aid in tracing the source of any errors. Functional and security testing is also performed, ensuring that the functional and security requirements are fulfilled. This research is relevant to the DoD Global Information Grid's vision of assured information sharing.

14. SUBJECT TERMS Monterey Security Architecture, WebDAV, Multilevel File Sharing			**15. NUMBER OF PAGES** 78
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

USE OF WEBDAV TO SUPPORT A VIRTUAL FILE SYSTEM IN A COALITION ENVIRONMENT

Jeremiah A. Bradney
Civilian, Naval Postgraduate School
B.A., Point Loma Nazarene University, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2006

Author: Jeremiah A. Bradney

Approved by: Cynthia E. Irvine, Ph.D.
Thesis Advisor

Thuy D. Nguyen
Co-Advisor

Peter J. Denning, Ph.D.
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Monterey Security Architecture (MYSEA) combines untrusted commercial-off-the-shelf components with specialized high-assurance trusted components to provide a trusted multilevel secure environment. However, MYSEA currently lacks a means of providing remote access to data on the MYSEA server, a vital service for users in any client-server environment. Access and interaction with both public and private server-resident data that is constrained by the underlying discretionary and mandatory access control policy enforcement mechanisms of the MYSEA server is required.

By enabling the use of WebDAV in MYSEA, this thesis provides a means for fulfilling the above requirement for secure remote access by creating a virtual web-based file system accessible from the MYSEA MLS network. This is accomplished by incorporating the mod_dav module into the Apache web server already installed on the MYSEA server. The use of a module required minimal changes to add the desired functionality. Integration of mod_dav is performed in three stages to aid in tracing the source of any errors. Functional and security testing is also performed, ensuring that the functional and security requirements are fulfilled. This research is relevant to the DoD Global Information Grid's vision of assured information sharing.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my advisors, Dr. Cynthia Irvine and Prof. Thuy Nguyen, for the time, effort, and guidance they have provided throughout this project. I would also like to thank Jean Khosalim and David Shifflett for their technical expertise and assistance.

This material is based upon work supported by the National Science Foundation under Grant No. DUE0414102. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

THIS PAGE INTENTIONALLY LEFT BLANK

ACRONYMS AND ABBREVIATIONS

CGI	Common Gateway Interface
COTS	Commercial Off The Shelf
DAC	Discretionary Access Control
DAV	Distributed Authoring and Versioning
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IMAP	Internet Message Access Protocol
MAC	Mandatory Access Control
MILS	Multiple Independent Levels of Security
MLS	Multilevel Secure
MYSEA	Monterey Security Architecture
OS	Operating System
RFC	Request For Comments
SAK	Secure Attention Key
SMTP	Simple Mail Transfer Protocol
STOP	Secure Trusted Operating Program
TCBE	Trusted Computing Base Extension
TPE	Trusted Path Extension
TSE	Trusted Services Engine

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

The balance between providing access to data for those who need it, and protecting that same data from unauthorized access is a difficult problem as well as a major concern for project managers who must deal with classified information. Multilevel Secure (MLS) systems provide a great deal of access restriction and, hopefully, assurance that only those who have authorization to access the data can do so. However, with this security often comes the sacrifice of usability, much to the chagrin of users and administrators alike. Unfortunately, it has thus far proved too difficult to provide both an adequate assurance of security as well as a user-friendly environment in one operating system.

By using a more user-friendly operating system to remotely access data located on the machine controlled by the more secure MLS operating system, it is possible to provide users with access to the functions that are restricted by the MLS server but are helpful for productivity. Security can also be maintained in this situation if measures are taken to ensure that unauthorized downgrading of information is prevented. These features are planned for the MYSEA client machines.

The motivation for this study is to use WebDAV to facilitate more flexible access to sensitive data in the MYSEA environment while maintaining the security of the MLS constraints.

B. PURPOSE

The objective of this project is to utilize the mod_dav Apache module in the existing installation of Apache on the MYSEA server to provide remote users on the MLS network a virtual file system with access to both private and public data stored on the MYSEA server.

C. ORGANIZATION OF PAPER

This thesis is organized as follows:

- Chapter I provides an introduction to the motivation and purpose of this thesis.

- Chapter II provides background information on WebDAV, the mod_dav Apache module and the MYSEA project.
- Chapter III describes the goals of the thesis, the concept of operation and the steps taken to complete the goal.
- Chapter IV describes the test plans used to validate each implementation phase of this thesis.
- Chapter V concludes with a project summary, a discussion or related work and suggestions for future work.

II. BACKGROUND

This chapter provides background information on various subjects related to the integration of WebDAV into the MYSEA MLS environment. The first section covers the WebDAV protocol extensions and uses as well as a specific implementation of the protocol while the second section introduces the MYSEA project.

A. WEBDAV

Web-based Distributed Authoring and Versioning (WebDAV) is a set of extensions to the Hypertext Transfer Protocol (HTTP). It is described in RFC 2518 [1] and developed by an Internet Engineering Task Force (IETF) working group that is aimed at allowing users to collaboratively edit and manage files on a remote web server. The goals of the WebDAV working group, as stated in their charter, are to "define the HTTP extensions necessary to enable distributed web authoring tools to be broadly interoperable, while supporting user needs." [2] In essence, WebDAV provides the capability to not only read static objects over a network, but to perform more interactive operations as well. The operations, or methods, included in WebDAV and a description of their functions can be referenced in Table 1.

Method	Description
PROPFIND	This method is used to retrieve the properties of a resource from the server.
PROPPATCH	This method is used to set the properties of a resource and propagate them to the server.
MKCOL	This method is used to create a new directory, known here as a collection.
GET	Essentially the same as the HTTP GET method, but extended for use with collections.
HEAD	Essentially the same as the HTTP HEAD method, but extended for use with collections.
POST	Same as the HTTP POST method. Includes all resources except for collections.
DELETE	This method deletes a resource, including collections.
PUT	Creates a new (non-collection) resource, or replaces and existing one, with data sent to the server.
COPY	Creates a duplicate of the target resource, including collections.
MOVE	Moves the target resource from its original location to a new location, including collections.
LOCK	Makes the specified property of a resource unable to be changed, including collections.
UNLOCK	Allows the specified property of a resource to be changed, including collections.

Table 1. WebDAV Methods

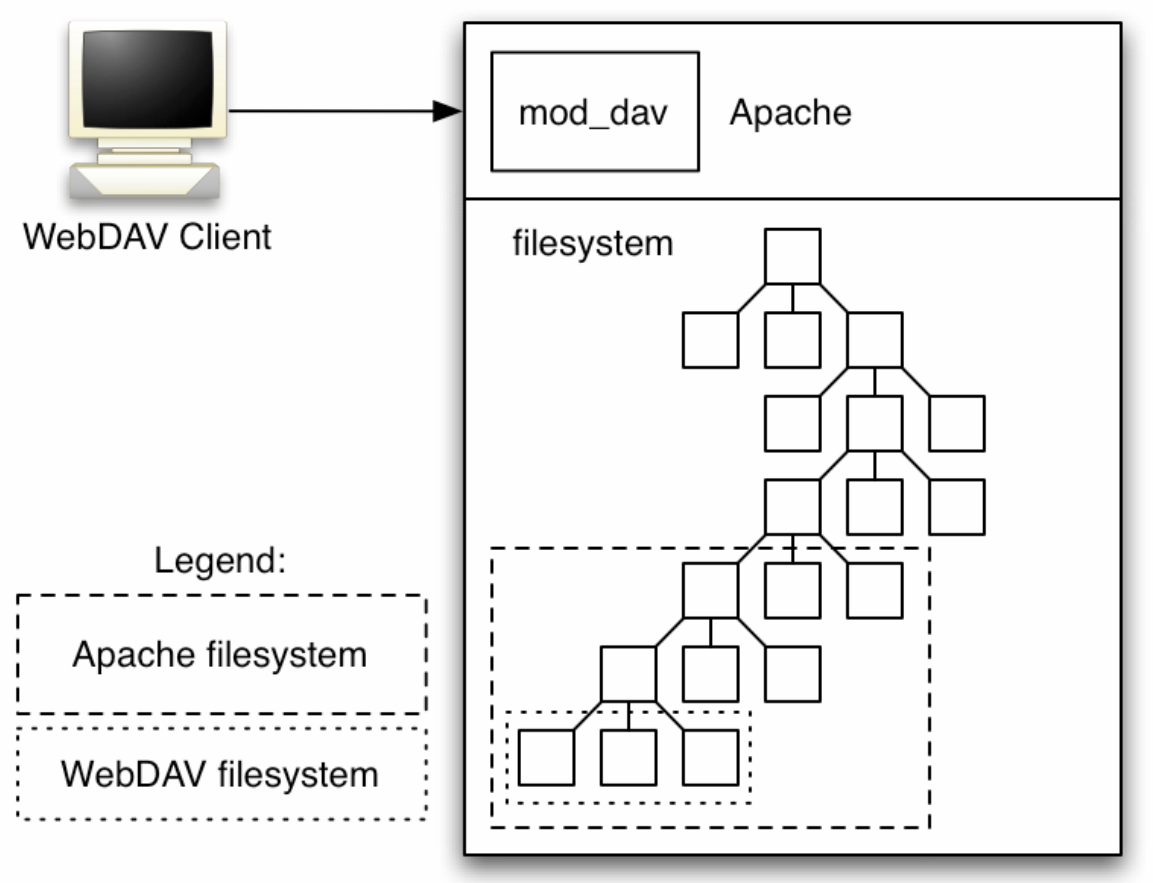

Figure 1. WebDAV Client-Server Architecture

Resources, collections and properties are the three major components of a WebDAV file system. Resources are any type of file, including collections. Collections are simply another word for directories. Properties are, to quote the RFC, “Properties are pieces of data that describe the state of a resource. Properties are data about data.” [1] For example, properties can contain basic information, such as subject and author, about a given resource. Properties are embodied through the use of specialized XML tags and can be controlled by both the server and the client.

1. WebDAV File System

An obvious use of these methods is to support a file system accessible from remote clients, as is shown in Figure 1. WebDAV enabled access to this file system would allow users to create, modify and share resources, within the specified WebDAV directory, remotely over a network. This would allow for both private file storage and

collaborative file repositories. The root directory of a WebDAV directory tree can coincide with an existing directory subtree, or can be isolated to its own section of the existing file system.

2. mod_dav Apache Module

The WebDAV module for the Apache webserver (named mod_dav) is one of the most common server-side implementations of the WebDAV protocol and is the implementation used in this thesis. Apache uses mod_dav to extend its capabilities to include those specified by the WebDAV protocol. The mod_dav module allows access to individual subtrees to be controlled on a per user basis with an access control list (ACL) determined in advance by the administrator via the standard configuration file. [3]

B. MYSEA

The Monterey Security Architecture (MYSEA) project is aimed at developing a secure architecture that will allow services and data from various networks of differing classification levels to be aggregated into a single, multilevel network. This is accomplished using commercial-off-the-shelf (COTS) components for performing most applications along with secure high assurance components that enforce the security policies. The use of COTS resources is desirable since COTS products are usually less expensive, more easily obtained and already in more widespread use than custom proprietary hardware and software. Unfortunately, COTS products rarely have adequate assurance that the policy they are designed to enforce will be done so sufficiently. This problem is solved in MYSEA through the incorporation of high assurance components. Thus in MYSEA, security is still enforceable across various sensitivity levels without needing to have physically separate hardware for each level of security. MYSEA relies on a high assurance platform that provides an adequate degree of certainty that the MLS policies will be enforced. MYSEA is aimed at achieving a higher adoption rate than previous MLS systems in organizations that already own a large number of COTS resources and do not wish to incur the cost of replacing the hardware and training (or retraining) their users on the new systems. [4][5]

Figure 2 is an illustration of the MYSEA network architecture. Communication between an untrusted client on the MLS LAN and various application services on either

the server or any of the connected networks is handled by an XTS-400 high assurance MLS server. This MLS server allows communication only at or below the currently negotiated session level. The security level of each session is negotiated with the MLS server via a Trusted Path Extension (TPE) that resides inline between the client and the server and establishes a trusted path, which may be encrypted, to the MLS server.

Currently, the MYSEA server supports HTTP, SMTP and IMAP. Tarantella (a web-portal for remotely running graphical applications, similar to X or Citrix), which executes on servers located on attached single level networks, is also available. These services allow for a wide variety of common applications to be used, but there is one major component lacking: the ability to remotely access and modify files in an MLS context. The goal of this thesis is to fill the gap by providing such a service through the use of WebDAV via a module extension to the existing Apache web server within the trusted, high assurance MYSEA server.

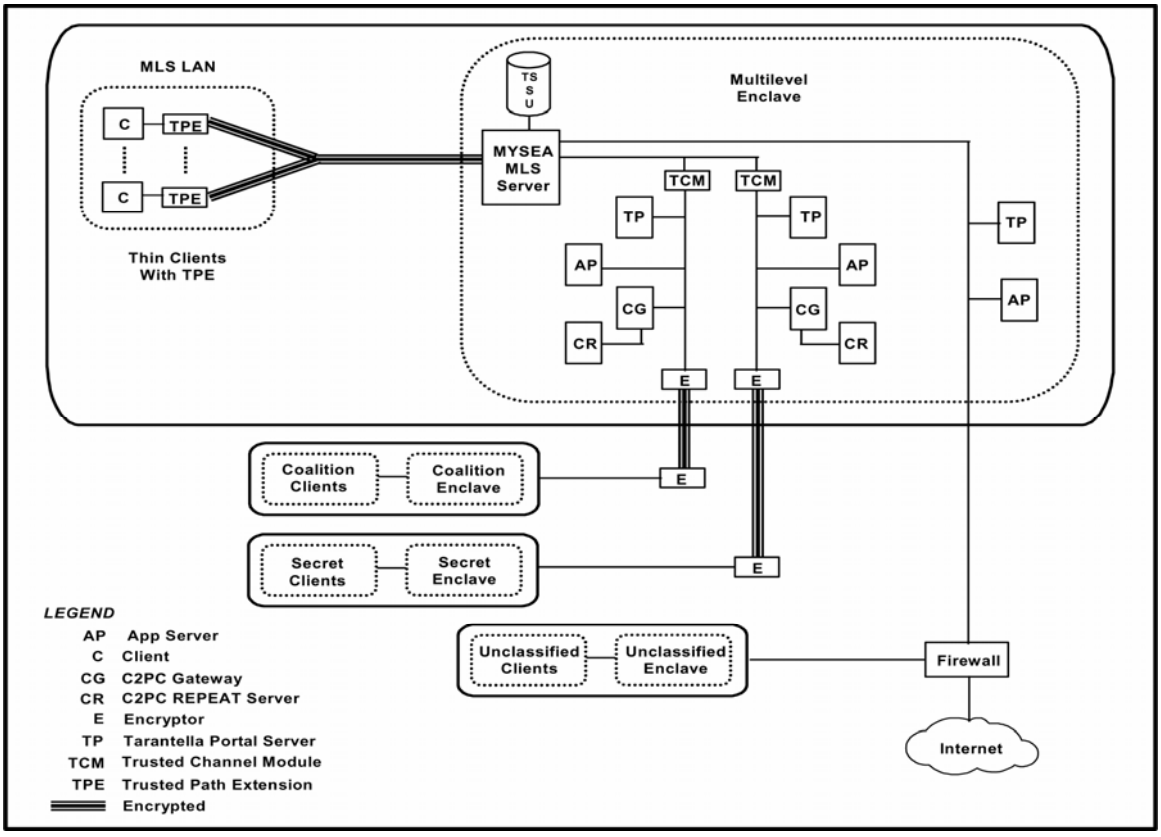

Figure 2. Testbed Topology [From Ref. 4]

C. SUMMARY

This chapter provided an introduction to WebDAV, gave brief background information on the MYSEA project and its related components and discussed why this thesis is needed. The next chapter will discuss the strategies used to get WebDAV running in different modes of operation on both Linux and the MYSEA server.

THIS PAGE INTENTIONALLY LEFT BLANK

III. PROJECT DESCRIPTION

This chapter will state the project goals, explain the concept of operation, describe the details of the hierarchical file system design and describe the porting methodology. The problems encountered during installation, as well as the steps taken to solve them, will be discussed along with the methodology.

A. GOALS

The goal of this thesis is to utilize WebDAV in the MYSEA multilevel environment to provide users on the MLS network with remote access to data whose security classification is dominated by the security level of the currently authorized session. Access to files is mediated by both the MAC and DAC policies of the server. WebDAV provides facilities for reading, writing and locking files, which can be useful for implementing a virtual network file system consisting of both a public file repository and private data stores. Ultimately, the use of WebDAV allows users with a WebDAV client to access data on the MYSEA server remotely, possibly in a similar manner to mounting and accessing data with a standard network drive. (Detailed instructions for using various clients to access a WebDAV directory can be found in Appendix A.) This provides a facility for data to be conveniently and securely accessed remotely in the MYSEA environment. Currently access to data on a MYSEA server is restricted to either browsing static web pages or through various CGI scripts that allow limited operations [6]. This project is intended to fill the need for more flexible access to data. A further requirement is that the client be able to “read down” to data at a lower security level to take advantage of the MLS capabilities offered by the server as well as to facilitate effective use by the MYSEA clients planned for the future [4][5].

B. CONCEPT OF OPERATION

In the MYSEA MLS network environment it is necessary for a client to authenticate to the MYSEA server before connecting to any service. Authentication can be achieved using either a TPE (Trusted Path Extension) device or the cross platform

TCBE (Trusted Computing Base Extension) Java client. With either of these methods a user can authenticate himself to the server and negotiate a session at a security level for which the user is authorized. The level of the negotiated session determines what level of access the user will have with any of the services on the MYSEA server, including WebDAV. This, along with the binding between a user's ID and subjects executing on his behalf, allows the underlying STOP 6.1 operating system to be used to enforce both MAC and DAC security policies.

Once the user is logged in and the session level is selected, a WebDAV client may be used to access files and perform operations (such as read, write, lock and unlock) available via the WebDAV server. Instructions for using various WebDAV clients can be found in Appendix A.

Once connected to the MYSEA WebDAV server, a user can access both private and public data at a level equal to or below the current session level. Whereas the private directories are available only to the user that owns them, the public directories are accessible by any user able to establish a session at the appropriate security level. The only other restriction is that files that are locked by a user cannot be overwritten or modified in any way until the lock is released.

The following two example scenarios illustrate possible collaboration between two users via WebDAV in a multilevel environment.

Scenario 1: Alice and Bob are logged in at the same security level. They can both simultaneously edit multiple files in the WebDAV directory, both in the public area and their own private directories. Alice decides she wants to edit the file "readme.txt" in the public directory, so she first locks the file to make sure that Bob cannot modify the same file while she is working on it. When Bob tries to edit "readme.txt", he will be able to open the file, but will be unable to save any changes because of the lock. However, Bob can save his edited copy of "readme.txt" to his private directory so that he can come back later to save his changes.

Scenario 2: Alice and Bob are logged in at different security levels; with Bob's level dominating Alice's. Alice still edits the "readme.txt" file, which is at the same level as Alice's established session level, but this time she forgets to lock it first. Bob can still

open up, read and edit “readme.txt”, but he cannot save the changes since he is not allowed to write information down to a lower level, as per the MAC policy. However, when Bob saves his edited version of “readme.txt”, this time it will be saved at Bob’s session level.

Although the process of connecting to and using WebDAV resources from a remote client is fairly simple, the exact steps differ from one client to another. Most major operating systems include support for connecting to and working with WebDAV servers by default. In most cases, the process of connecting to a WebDAV server is very similar, if not identical, to connecting to and mounting a network drive. There is also a third party application called Cadaver [7] that runs on any UNIX based operating system. Cadaver acts like most command line FTP clients and allows for the use of a wider range of WebDAV functions (such as file locking) as compared to the graphical clients (i.e. the clients accessible through a GUI). Instructions for using built-in WebDAV clients (for Windows, Fedora Core 4 and Macintosh) as well as for using Cadaver are provided in Appendix A.

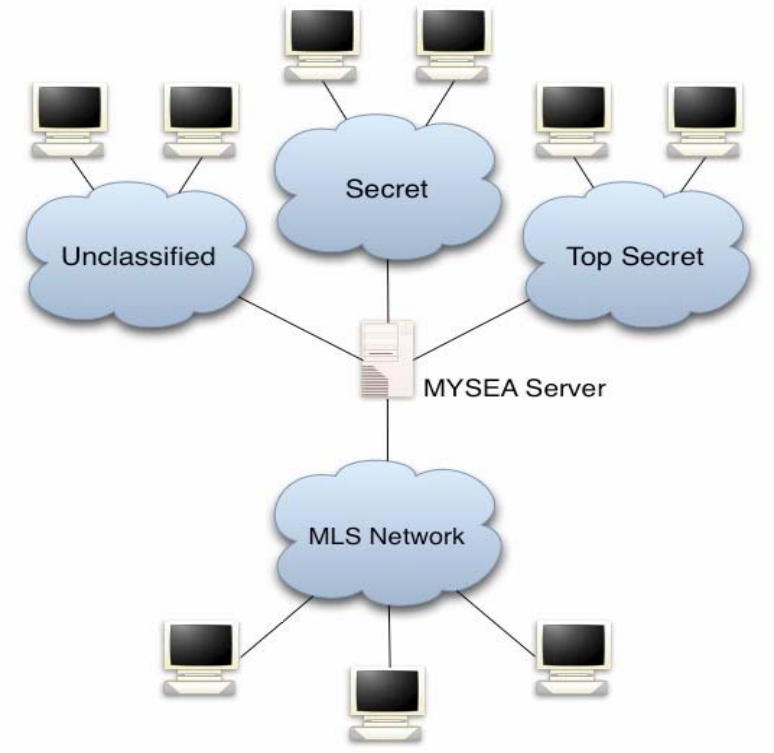

Figure 3. MYSEA Network Topology

The Apache server that is currently installed and configured on the MYSEA server is only accessible from the MLS network and not from any of the single level networks connected to the MYSEA server. It would be possible to set up an Apache server with WebDAV for each of the single level networks to allow access to the same set of data as is available to the MLS network. Since users accessing from a single level network cannot change their session level via a TPE or TCBE, access would be less flexible than from the MLS network since users on a single level network are restricted to the security level that network is bound to. A user on a single level network would still be able to read down to data at a lower security level, but would be unable to read anything higher or write anything other than at the level he or she is bound to, even though the user may be cleared for access to more sensitive information. See Figure 3 for an illustration of the network topology.

C. DESIGN

Currently, the file system on the XTS-400 is set up to keep most of the MYSEA-related applications (including the Apache httpd binary), configuration files and data in one directory, specifically "/usr/local/mysea/"[1]. However, Apache has been configured in MYSEA to use "/home/http/", the home directory of the http user, to store its various configuration files and data, while "/usr/local/mysea/apache/" contains the executable binary and related source code. The main configuration file is named and located at "/home/http/conf/httpd.conf" and the Document Root is located at "/home/http/htdocs/". The Document Root, by definition, contains all of the files that can be served by (i.e. accessed remotely through) the Apache server.

To add WebDAV support to the existing Apache server, a new directory, "/home/http/htdocs/dav/", was added to the Document Root. For the central public WebDAV directory, a new directory, "/home/http/htdocs/dav/public/", was created. To allow individual users to have a private data store available via WebDAV, a new directory, "/home/http/htdocs/dav/private/", was added to contain a symbolic link to each

[1] This and all directory paths given in this chapter are absolute (with respect to the root of the file system) rather than relative. All directories referenced are at *secrecy level 0* and *integrity level 3* unless otherwise noted.

user's home directory. The Apache configuration was then altered (as per the instructions in Appendix A) to allow WebDAV to be used in the "/home/http/htdocs/dav/" directory. This also allows all subdirectories of this directory, the *root DAV directory*, to be accessed with WebDAV. See Figure 4 for an illustration of the changes made to the file system.

For each directory in the XTS-400 file system that is set up for WebDAV use, both in the public and private sections, various subdirectories need to be created for each combination of secrecy and integrity levels that need to be accessed. These different single level directories need to be created administratively and cannot be added by the user via WebDAV. An alternative, simpler way to implement a similar scheme would be to use a deflection directory. However, associated with the deflection directory method are the undesirable consequences described below.

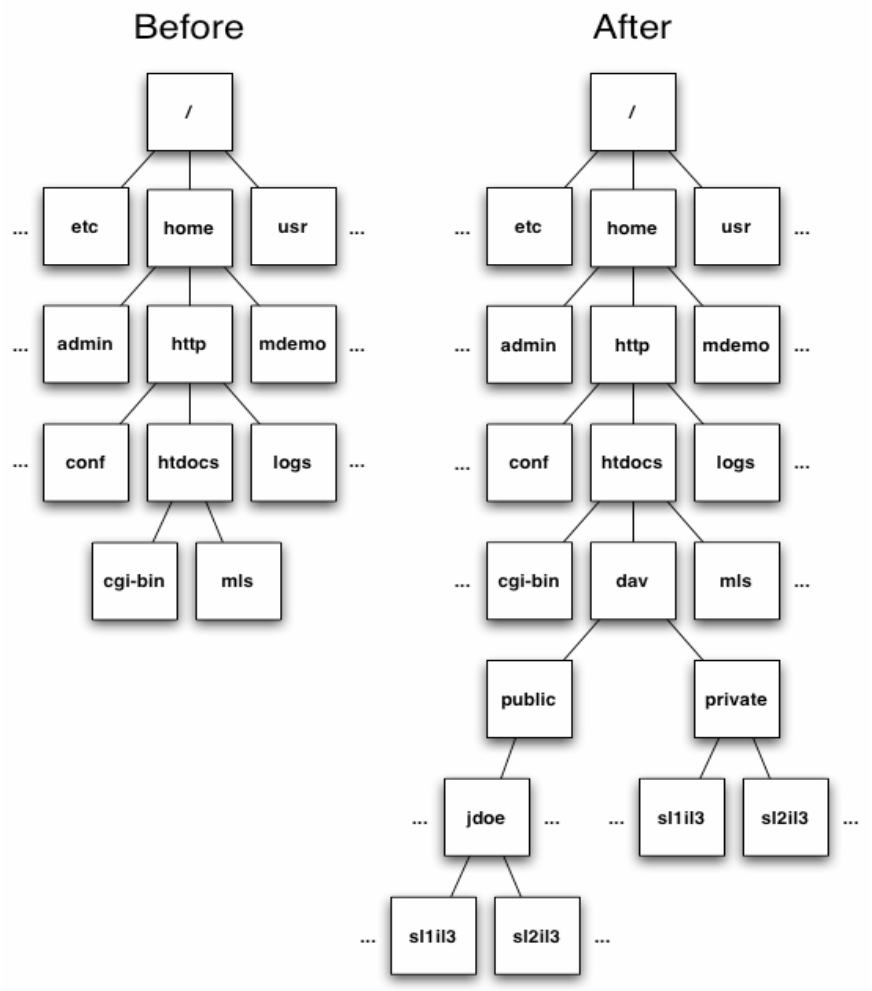

Figure 4. File System Before and After

A deflection directory is, in essence, a directory that exists independently at all security levels simultaneously. In the deflection model, when a directory is accessed an instance of the directory corresponding to the current security level is automatically chosen by the OS. There is no possibility of accessing a deflection directory at a level other than the level of the current session, and no data is shared between different instances. This ensures that no data can be read from a higher security level or written to a lower security level. However, the use of deflection directories also prevents data from being read from a security level lower than that of the current session. Since it is a MYSEA requirement that users can read data at a lower security level, this arrangement

was deemed unacceptable. This requirement led to the conclusion that using separate directories for each security level was a more desirable design than that of using a single deflection directory.

To keep the personal WebDAV directories private, their DAC properties were set to allow only the owner to have read and write access to these directories. Each user also needs to have an individual directory for each security level accessed. Since the user cannot create these directories at the required security levels over the WebDAV connection, it is necessary for the administrator to create, at the minimum, a select few directories at commonly used security levels at the time the user account is created.

D. METHODOLOGY

While it would have been possible to begin by implementing WebDAV in the multilevel MYSEA environment, as was ultimately the goal, it would have been much more difficult to trace any errors to Apache, mod_dav, the MYSEA services or the underlying STOP 6.1 operating system. Instead, the implementation was performed in stages, with testing conducted at each stage to ensure that the required functionality was preserved. Table 2 contains information about the platforms used in each stage.

The minimal testing performed at each stage consisted of connecting to the server and attempting to read and modify an existing file. This testing was done merely to ensure that the installation had been successful and was performed with the client on the machine running Linux. More thorough testing will be discussed in Chapter IV.

	Hardware	OS	Security Context	Apache Version	mod_dav Version
Stage 1	Dell Inspiron	Fedora Core 4	N/A	1.3.34	1.0.3
Stage 2	XTS-400	STOP 6.1	Single Level	1.3.34	1.0.3
Stage 3	XTS-400	STOP 6.1	Multilevel	1.3.34	1.0.3

Table 2. Stage Platform Information

1. Linux

For the first stage, Apache 1.3.34 [8] and mod_dav 1.0.3 [9] were downloaded and installed on a Dell laptop running Linux, specifically Fedora Core 4 [10]. Here the purpose was familiarization with the installation and use of WebDAV and Apache. Apache was chosen for use not only because it is among the most widely used web servers but because it is already installed and operating in the MYSEA testbed. Adding mod_dav, an Apache module to enable the use of WebDAV, to the existing Apache server seemed to be the most straightforward route to implementing WebDAV in the MYSEA environment. Using the documentation that accompanied the downloaded files, both Apache and mod_dav were compiled and installed without incident. After the successful completion of a few basic tests that demonstrated the operation of WebDAV in a standard environment it was decided to move on to the next step.

2. Single Level XTS

After familiarization with the XTS-400, the same versions of Apache and mod_dav as are discussed in the previous section were downloaded, installed and configured on the XTS-400 running the STOP 6.1 [11] operating system. For this phase, Apache was configured to run as a standalone server at a single classification level. The security level of this server needed to be *secrecy level 1* and *integrity level 3* since the network interface chosen for use is configured to be accessible only as this level. The only hurdles to this goal were those related to familiarization with the new operating system, specifically problems associated with properly starting the networking services. Since the installation and configuration were identical to those of the Linux system, it took very little time to set up Apache with mod_dav at a single level on the XTS-400. Some basic functionality tests were successfully performed; these ensured that the installation had been performed properly. After these encouraging results the next step was to install and configure mod_dav in the multilevel environment.

3. Multilevel XTS

For a single Apache server to be available at multiple levels, it must be set up as an inetd service, bound to a specific port and registered with the MYSEA daemon. The MYSEA services behave similarly to the classic inetd service in that they listen for new incoming connections and spawns new processes as appropriate to handle incoming

requests. The difference is that the MYSEA services will spawn the new process at the same security and integrity level as the negotiated level of the incoming request. However, for a service to work like this in MYSEA, application code must be modified to use a form of virtual sockets implemented in the MYSEA system rather than the standard socket implementation.

Since Apache 1.3.12 [12] had already been modified to use the virtual sockets, and since mod_dav is advertised in the documentation as being compatible with versions of Apache as old as 1.3.6 [3], an attempt was made to install mod_dav with the existing Apache installation. Unfortunately, mod_dav did not work with this setup. Apache still worked after installing mod_dav, but any attempt to use the new protocol methods specific to WebDAV was met with a 405 “Method Not Allowed” error.

To ensure that this error was not simply caused by the code modifications interfering with mod_dav, an unmodified version of Apache 1.3.12 was tested at a single level, as was successfully done with Apache 1.3.34 previously. However, the same error occurred. It was then decided to test an unmodified version of Apache 1.3.12 on Fedora Core 4. This was to ensure that the problem wasn’t associated with the XTS-400. Surprisingly, this attempt was met with the same error as before, even though mod_dav documentation claims that it works with Apache 1.3.12. Rather than spend more time attempting to determine the cause of the error, it was decided that the best course of action would be to make the appropriate modifications to Apache 1.3.34 in order to get it to function with the MYSEA services.

Once the MYSEA installation of Apache had been upgraded to version 1.3.34, another attempt was made to install and utilize mod_dav in the MYSEA environment. With only a slight modification to the installation procedure and configuration from the previous stages, mod_dav was successfully installed and run. Basic testing confirmed that mod_dav was indeed functioning correctly. Comprehensive testing was subsequently performed, the procedures and results of which are described in Chapter IV.

E. SUMMARY

This chapter explained the goals of this thesis, described the design of the file system structure, provided a concept of operation and detailed the steps taken to incorporate WebDAV in various environments. The next chapter will outline a test plan to ensure that both functional and security requirements are met as well as describe the details of the test results.

IV. TESTING AND RESULTS

This chapter will detail the functional and security tests for the use of WebDAV in various settings and capacities. Functional testing is required in order to compare the actual WebDAV functionality to the expected functionality. The functional tests confirm whether the functionality expected of the system is operable and provide a quantifiable measurement of the ability of the system to successfully perform the desired tasks. The tests performed fall into three categories: Linux, Single Level XTS and Multilevel XTS. Each of these categories corresponds to a stage as described in Table 2 in the previous chapter.

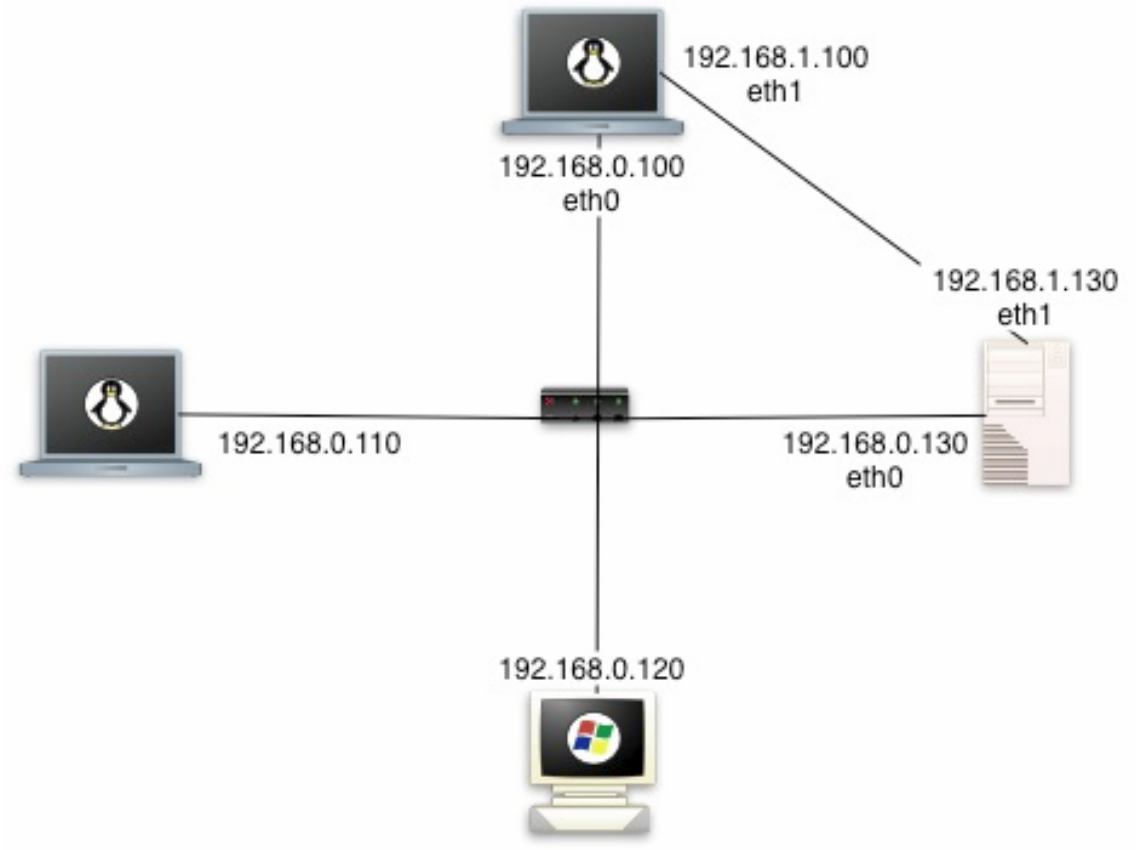

Figure 5. Test Network Topology

Security testing is required to ensure that the MAC and DAC policies are correctly enforced. Since the DAC policies on the Linux may be implemented differently than those on the XTS-400, only the two XTS stages were tested for security. The security tests performed fall into two categories: Single Level XTS and Multilevel XTS. In the multilevel category, the testing scope was expanded to encompass the added security parameter possibilities.

All testing was performed on a small network with five computers connected via a switch, as illustrated in Figure 5. The operating systems used on the five systems will be: Fedora Core 4, Windows XP, Mac OS X 10.3 and STOP 6.1 (as part of the XTS-400). The Windows, Mac and Linux systems were used as the clients, whereas a Linux system and the XTS-400 acted as the server in different implementation phases illustrated in Table 2 of Chapter III. In all three cases, the Windows computer, the Macintosh and at least one of the Linux computers were used as clients to test the WebDAV server. On the Windows and Macintosh machines, the built in WebDAV client was used. On the Linux clients, both the built in WebDAV client and an open source client, called Cadaver, were used. Simultaneous connections from multiple clients were tested with no issues.

A. FUNCTIONAL TEST PLAN

Tests were performed on both files and directories. The functions tested in each implementation were as follows: read, write, create, lock, unlock, delete, make directory, change directory and delete directory. These functions were tested with all four types of WebDAV clients.

The success or failure of a particular test is indicated with a simple pass/fail notation. The requirement for a “pass” mark is only that the desired outcome is achieved. All of these tests are expected to pass where the related function has been implemented in the client.

Test #	Test Type	Test Description
A1	Read	Test A1 ensures that files can be read via the WebDAV client.
A2	Write	Test A2 ensures that files can be written via the WebDAV client.
A3	Create	Test A3 ensures that files can be created via the WebDAV client.
A4	Lock	Test A4 ensures that files can be locked via the WebDAV client.
A5	Unlock	Test A5 ensures that files can be unlocked via the WebDAV client.
A6	Delete	Test A6 ensures that files can be deleted via the WebDAV client.
A7	Make Directory	Test A7 ensures that directories can be created via the WebDAV client.
A8	Change Directory	Test A8 ensures that directories can be traversed via the WebDAV client.
A9	Delete Directory	Test A9 ensures that directories can be deleted via the WebDAV client.
A10	Read Nonexistent File	Test A10 ensures that no unrecoverable errors occur when attempting to read a file that does not exist.
A11	Delete Nonexistent File	Test A11 ensures that no unrecoverable errors occur when attempting to delete a file that does not exist.
A12	Lock Nonexistent File	Test A12 ensures that no unrecoverable errors occur when attempting to lock a file that does not exist.
A13	Unlock Nonexistent File	Test A13 ensures that no unrecoverable errors occur when attempting to unlock a file that does not exist.
A14	Delete Nonexistent Directory	Test A14 ensures that no unrecoverable errors occur when attempting to delete a directory that does not exist.

Table 3. Functional Test Descriptions

B. LINUX TESTING

The functional tests performed using Linux as the server define a baseline of functionality desired for a WebDAV server. In other words, if a particular function did not work on the Linux implementation of the WebDAV server, then it is not expected to work on the XTS-400 implementation either.

1. Test Results

All of the clients tested performed perfectly, with one exception. The exception is that all of the graphical clients, the ones that were built into the various operating systems, had no facilities for locking or unlocking files or directories, or interacting with nonexistent files or directories. Cadaver, however, was able to lock and unlock both files and directories with ease. Files that were locked with Cadaver were unable to be

overwritten by any client, but they could still be read (as should be the case). Test results marked "N/A" were those that cannot be performed with graphical clients, as noted above.

Test #	Test Type	Windows Explorer	Macintosh Finder	Linux Gnome	Cadaver
A1	Read	Pass	Pass	Pass	Pass
A2	Write	Pass	Pass	Pass	Pass
A3	Create	Pass	Pass	Pass	Pass
A3	Lock	N/A	N/A	N/A	Pass
A4	Unlock	N/A	N/A	N/A	Pass
A5	Delete	Pass	Pass	Pass	Pass
A6	Make Directory	Pass	Pass	Pass	Pass
A7	Change Directory	Pass	Pass	Pass	Pass
A8	Delete Directory	Pass	Pass	Pass	Pass
A9	Read Nonexistent File	N/A	N/A	N/A	Pass
A10	Delete Nonexistent File	N/A	N/A	N/A	Pass
A11	Lock Nonexistent File	N/A	N/A	N/A	Pass
A12	Unlock Nonexistent File	N/A	N/A	N/A	Pass
A13	Delete Nonexistent Directory	N/A	N/A	N/A	Pass

Table 4. Linux Server Functional Test Results

C. SINGLE LEVEL XTS TESTING

The tests described in this section were performed on the XTS-400 and were designed to test that the relevant MAC security constraints are met by the implementation and that no functionality was lost in the transition from the Linux server to the XTS-400 single level implementation.

1. Security Test Plan

The set of tests described in Table 5 were designed to determine whether or not the MAC secrecy constraints are adequately enforced over a WebDAV connection. Since this implementation of the server is bound to a single security level, only variance in the security levels of the objects being accessed can be tested.

Test #	Test Type	Session Level	Object Level	Operation	Expected Result
B1	Secrecy Read Up	sl1:il3	sl2:il3	Read	Fail
	Test B1 ensures that a user is not able to read data from a secrecy level higher than the level of the currently established session.				
B2	Secrecy Read Down	sl1:il3	sl0:il3	Read	Pass
	Test B2 ensures that a user is able to read data from a secrecy level lower than the level of the currently established session.				
B3	Secrecy Read Equal	sl1:il3	sl1:il3	Read	Pass
	Test B3 ensures that a user is able to read data from a secrecy level equal to the level of the currently established session.				
B4	Secrecy Write Up	sl1:il3	sl2:il3	Write	Fail
	Test B4 ensures that a user is not able to write data to a secrecy level higher than the level of the currently established session.				
B5	Secrecy Write Down	sl1:il3	sl0:il3	Write	Fail
	Test B5 ensures that a user is not able to write data to a secrecy level lower than the level of the currently established session.				
B6	Secrecy Write Equal	sl1:il3	sl1:il3	Write	Pass
	Test B6 ensures that a user is able to write data to a secrecy level equal to the level of the currently established session.				

Table 5. Single Level Security Tests

2. Test Results

As Table 6 shows, the results of the functional tests were the same as those performed with the Linux server. This indicates that no functionality was changed during the transition to the XTS-400.

Test #	Test Type	Windows Explorer	Macintosh Finder	Linux Gnome	Cadaver
A1	Read	Pass	Pass	Pass	Pass
A2	Write	Pass	Pass	Pass	Pass
A3	Create	Pass	Pass	Pass	Pass
A3	Lock	N/A	N/A	N/A	Pass
A4	Unlock	N/A	N/A	N/A	Pass
A5	Delete	Pass	Pass	Pass	Pass
A6	Make Directory	Pass	Pass	Pass	Pass
A7	Change Directory	Pass	Pass	Pass	Pass
A8	Delete Directory	Pass	Pass	Pass	Pass
A9	Read Nonexistent File	N/A	N/A	N/A	Pass
A10	Delete Nonexistent File	N/A	N/A	N/A	Pass
A11	Lock Nonexistent File	N/A	N/A	N/A	Pass
A12	Unlock Nonexistent File	N/A	N/A	N/A	Pass
A13	Delete Nonexistent Directory	N/A	N/A	N/A	Pass

Table 6. Single Level XTS Server Functional Test Results

The results in Table 7 show that all of the tested clients interacted with the server as expected when faced with MAC secrecy constraints of the types shown. These results are encouraging and show that, at least for basic secrecy interactions, the MAC policies are properly enforced over WebDAV.

Test #	Test Type	Windows	Macintosh	Linux	Cadaver
B1	Secrecy Read Up	Fail	Fail	Fail	Fail
B2	Secrecy Read Down	Pass	Pass	Pass	Pass
B3	Secrecy Read Equal	Pass	Pass	Pass	Pass
B4	Secrecy Write Up	Fail	Fail	Fail	Fail
B5	Secrecy Write Down	Fail	Fail	Fail	Fail
B6	Secrecy Write Equal	Pass	Pass	Pass	Pass

Table 7. Single Level XTS Server Security Test Results

D. MULTILEVEL XTS TESTING

The tests described in this section were performed on the XTS-400 and were designed to test that the relevant MAC security constraints are met by the implementation and that no functionality was lost in the transition from the single level to the multilevel implementation on the XTS-400.

1. Security Test Plan

The set of tests described in Table 8 were designed to determine whether or not the MAC and DAC security constraints are adequately enforced over a WebDAV connection. This implementation of the server is able to operate with sessions established at different secrecy and integrity levels, so appropriate tests that varied the secrecy and integrity levels of both the session and the objects were performed.

Since many different users can establish a session with the server, tests were also included to ensure that the DAC policies were also adhered to. These tests were performed with the owner of the session and the owner of the object differing from each other. The secrecy levels of the objects were also varied to make sure a “high” user can not access the files of a different “low” user, and vice versa, in an unauthorized manner.

Test #	Test Type	Session Level	Object Level	Operation	Expected Result
C1	Secrecy Read Up	sl1:il3	sl2:il3	Read	Fail
C2		sl2:il3	sl3:il3	Read	Fail
C3		sl3:il3	sl4:il3	Read	Fail
	Tests C1 through C3 ensure that a user is not able to read data from a secrecy level higher than the level of the currently established session.				
C4	Secrecy Read Down	sl1:il3	sl0:il3	Read	Pass
C5		sl2:il3	sl1:il3	Read	Pass
C6		sl3:il3	sl2:il3	Read	Pass
	Tests C4 through C6 ensure that a user is able to read data from a secrecy level lower than the level of the currently established session.				
C7	Secrecy Read Equal	sl1:il3	sl1:il3	Read	Pass
C8		sl2:il3	sl2:il3	Read	Pass
C9		sl3:il3	sl3:il3	Read	Pass
	Tests C7 through C9 ensure that a user is able to read data from a secrecy level equal to the level of the currently established session.				
C10	Secrecy Write Up	sl1:il3	sl2:il3	Write	Fail
C11		sl2:il3	sl3:il3	Write	Fail
C12		sl3:il3	sl4:il3	Write	Fail
	Tests C10 through C12 ensure that a user is not able to write data to a secrecy level higher than the level of the currently established session.				
C13	Secrecy Write Down	sl1:il3	sl0:il3	Write	Fail
C14		sl2:il3	sl1:il3	Write	Fail
C15		sl3:il3	sl2:il3	Write	Fail
	Tests C13 through C15 ensure that a user is not able to write data to a secrecy level lower than the level of the currently established session.				
C16	Secrecy Write Equal	sl1:il3	sl1:il3	Write	Pass
C17		sl2:il3	sl2:il3	Write	Pass
C18		sl3:il3	sl3:il3	Write	Pass
	Tests C16 through C18 ensure that a user is able to write data to a secrecy level equal to the level of the currently established session.				
C19	DAC Read Down	sl1:il3	sl0:il3	Read	Fail
	Test C19 ensures that a user is not able to read data from a file that is owned by another user, has permissions set to allow only the owner to read and has a secrecy level lower than the level of the currently established session.				
C20	DAC Read Equal	sl1:il3	sl1:il3	Read	Fail
	Test C20 ensures that a user is not able to read data from a file that is owned by another user, has permissions set to allow only the owner to read and has a secrecy level equal to the level of the currently established session.				
C21	DAC Write Equal	sl1:il3	sl1:il3	Write	Fail
	Test C21 ensures that a user is not able to write data to a file that is owned by another user, has permissions set to allow only the owner to write and has a secrecy level equal to the level of the currently established session.				
C22	Multiple Lock	sl1:il3	sl1:il3	Lock	Fail
	Test C22 ensures that multiple users are not able to lock the same file simultaneously.				
C23	Multiple Lock/Write	sl1:il3	sl1:il3	Lock/Write	Fail
	Test C23 ensures that a user is not able to write to a file that has been locked by another user.				

Table 8. Multilevel Security Tests

2. Test Results

As Table 9 shows, the results of the functional tests were the same as the results of the tests performed with the Linux server and the server running on the XTS-400 at a single level. This indicates that no functionality was changed during the transition to the multilevel XTS-400 environment.

Test #	Test Type	Windows Explorer	Macintosh Finder	Linux Gnome	Cadaver
A1	Read	Pass	Pass	Pass	Pass
A2	Write	Pass	Pass	Pass	Pass
A3	Create	Pass	Pass	Pass	Pass
A3	Lock	N/A	N/A	N/A	Pass
A4	Unlock	N/A	N/A	N/A	Pass
A5	Delete	Pass	Pass	Pass	Pass
A6	Make Directory	Pass	Pass	Pass	Pass
A7	Change Directory	Pass	Pass	Pass	Pass
A8	Delete Directory	Pass	Pass	Pass	Pass
A9	Read Nonexistent File	N/A	N/A	N/A	Pass
A10	Delete Nonexistent File	N/A	N/A	N/A	Pass
A11	Lock Nonexistent File	N/A	N/A	N/A	Pass
A12	Unlock Nonexistent File	N/A	N/A	N/A	Pass
A13	Delete Nonexistent Directory	N/A	N/A	N/A	Pass

Table 9. Multilevel XTS Server Functional Test Results

The results in Table 10 show that all of the tested clients interacted with the server as expected when faced with MAC and DAC constraints of the types shown. These results are encouraging and show that even for complex security interactions, the MAC and DAC policies are properly enforced over WebDAV.

Test #	Test Type	Windows	Macintosh	Linux	Cadaver
C1	Secrecy Read Up	Fail	Fail	Fail	Fail
C2		Fail	Fail	Fail	Fail
C3		Fail	Fail	Fail	Fail
C4	Secrecy Read Down	Pass	Pass	Pass	Pass
C5		Pass	Pass	Pass	Pass
C6		Pass	Pass	Pass	Pass
C7	Secrecy Read Equal	Pass	Pass	Pass	Pass
C8		Pass	Pass	Pass	Pass
C9		Pass	Pass	Pass	Pass
C10	Secrecy Write Up	Fail	Fail	Fail	Fail
C11		Fail	Fail	Fail	Fail
C12		Fail	Fail	Fail	Fail
C13	Secrecy Write Down	Fail	Fail	Fail	Fail
C14		Fail	Fail	Fail	Fail
C15		Fail	Fail	Fail	Fail
C16	Secrecy Write Equal	Pass	Pass	Pass	Pass
C17		Pass	Pass	Pass	Pass
C18		Pass	Pass	Pass	Pass
C19	DAC Read Down	Fail	Fail	Fail	Fail
C20	DAC Read Equal	Fail	Fail	Fail	Fail
C21	DAC Write Equal	Fail	Fail	Fail	Fail
C22	Multiple Lock	N/A	N/A	N/A	Fail
C23	Multiple Lock/Write	N/A	N/A	N/A	Fail

Table 10. Multilevel XTS Server Security Test Results

E. PROBLEMS ENCOUNTERED

There was only one major problem encountered during the testing of the WebDAV servers, and it occurred only with the instance installed in the multilevel environment. When accessing one of the directories that contained subdirectories with varying security levels, the "/home/http/htdocs/dav/public/" directory for example, any attempt to list the contents of the directory via WebDAV failed unless the current session was at the same level as the highest level subdirectory present. For example, say a directory has three subdirectories at the levels *sl0:il3*, *sl1:il3* and *sl2:il3*. If a user with a session established at *sl1:il3* attempts to list the contents of this directory with WebDAV, the operation will fail. If another user with a session established at *s12:il3* attempts to list the contents of this directory with WebDAV, the operation will succeed.

In order to fix this problem, the source code for mod_dav will likely need to be edited. It is suspected that the failure stems from the use of a stat() function to determine the properties of the contents of the directories. The stat() function is likely to fail when

attempted on a directory with a level higher than that of the current session. Since mod_dav is not multilevel aware and does not expect the stat() function to fail, mod_dav returns an error rather than a partial directory listing.

This error should be possible to fix with minimum effort, pinpointing the exact cause, however, could prove difficult. It has been decided that rectifying this error is beyond the scope of this thesis.

F. SUMMARY

This chapter explained both the functional and security tests performed on three instances of the WebDAV server: on Linux, on the XTS-400 at a single level and on the XTS-400 at multiple levels. In addition the results of the tests were explained. The next chapter will discuss the results and conclusions of the work performed and will suggest possible future work in related areas.

V. CONCLUSION AND FUTURE WORK

This chapter will state the project conclusions, discuss a related project and suggest possible work for the future.

A. CONCLUSION

The goal of this thesis is to utilize WebDAV in the MYSEA multilevel environment to provide users on the MYSEA MLS network with remote access to data whose security classification is dominated by the security level of the currently authorized session. This goal has been successfully achieved through the use of Apache and the mod_dav module on the MYSEA server. The tests described in Chapter IV confirmed that the WebDAV server is fully functional (with the only limitations being in the functions implemented in some of the clients) and that it adequately enforces the MAC and DAC policies of the MYSEA server.

B. RELATED WORK

Galois Connections, Inc. has a cross-domain solutions project called the Trusted Services Engine (TSE) that utilizes WebDAV to implement a MILS file system that is remotely accessible over a network. [13] The major difference between the TSE and the way WebDAV is used in the MYSEA environment is that the TSE is built on a MILS kernel that requires a separate hard disk and an independent instance of each server at each security level in order to provide multilevel services. This approach does not scale well to arbitrary numbers of security levels since each additional level would require its own hard disk and another instance of each required software component. The MYSEA server is trusted to provide multilevel protection while using only one shared file system. It also automatically spawns new instances of each server process at the appropriate security level as new connections are established.

C. FUTURE WORK

There were two major issues that arose through the work on this thesis that warrant further work.

1. Account Setup

The first issue is the amount of effort required by an administrator to set up individual user accounts for use with WebDAV. As it stands, an administrator must manually create various directories at the proper security levels for each user that requires access to private data via WebDAV. Creating and setting these directories to various security levels is a time consuming process. However, this process can be automated.

2. Directory Listing

The second issue is the inability to list the contents of directories that contain items at different security levels. Although this problem is mostly an inconvenience, it is most likely possible to modify the source code for the mod_dav module in such a way as to overcome this error.

3. Expanded WebDAV Access

As discussed in Section B of Chapter III, the third future work item is to set up separate instances of Apache with mod_dav on the MYSEA server, one for each of the connected single level networks. Each of these instances would share the same document root, which could also coincide with the document root used by the server in this thesis, thus allowing more authorized users access to the same shared data at multiple levels while still allowing for “read down” capabilities.

APPENDIX A: INSTALLATION PROCEDURES

This appendix describes the installation procedures for the Apache 1.3.34 web server, the mod_dav 1.0.3 Apache module and the Cadaver 0.22.2 WebDAV client. Instructions for testing the installation and using various WebDAV clients are also provided. Each section is independent and does not depend of the steps performed in any other section. The only exception is that the testing procedures cannot be performed until at least one of the mod_dav installations had occurred. Each set of instructions will explicitly state which operating system the steps are to be performed on.

The following instructions also make reference to the Secure Attention Key (SAK). On the XTS-400, the SAK is invoked by simultaneously pressing both the "Alt" and "SysRq" (a.k.a. "Print Screen") keys. This allows special trusted commands to be performed. Of special note are two trusted commands: the *sl* command and the *fsm* command. The *sl* command is used to set the security level of a session while the *fsm* command is used to view and edit various properties (i.e. the MAC and DAC permissions) of files and directories.

A. INSTALLATION AND TEST TOPOLOGY

The network topology used for this installation and testing activity consists of four computers connected via a switch as shown in Figure 5 of Chapter IV.. Of these four computers, two run Fedora Core 4, one Windows XP and the last is an XTS-400 running the STOP 6.1 operating system.

B. LINUX INSTALLATION

All of the procedures in this section are to be performed as the root user on a machine running Fedora Core 4. These procedures download, install and configure the Apache 1.3.34 web server along with the mod_dav 1.0.3 Apache module.

Step 1. Download and unpack the Apache server and the mod_dav module from the installation CD:

```
cd /root
cp /media/cdrom/apache_1.3.34.tar.gz ./
cp /media/cdrom/mod_dav-1.0.3-1.3.6.tar.gz ./
gunzip apache_1.3.34.tar.gz
```

```
tar -xvf apache_1.3.34.tar
gunzip mod_dav-1.0.3-1.3.6.tar.gz
tar -xvf mod_dav-1.0.3-1.3.6.tar
```

Step 2. Build and install mod_dav into the Apache source directory. While this is done before Apache is built and installed, the Apache configuration script must be run first to ensure that certain files required for mod_dav configuration are present.

```
cd /root/apache_1.3.34
./configure --prefix=/usr/local/apache13
cd /root/mod_dav-1.0.3-1.3.6
./configure --with-apache=/root/apache_1.3.34
make
make install
```

Step 3. Build and install Apache:

```
cd /root/apache_1.3.34
./configure --prefix=/usr/local/apache13 --activate-
     module=src/modules/dav/libdav.a
make
make install
```

Step 4. Save and edit the default Apache configuration:

```
cd /usr/local/apache13/conf/
cp httpd.conf httpd.conf.bak
vi httpd.conf
```

Step 5. Append the following text to the end of the “httpd.conf” file:

```
DAVLockDB /usr/local/apache13/var/DAVLock
<Location /dav>
     DAV On
</Location>
```

Step 6. Create the two directories referenced in the configuration lines above:

```
mkdir /usr/local/apache13/var
mkdir /usr/local/apache13/htdocs/dav
/usr/local/apache13/bin/apachectl start
```

Once all of these steps have been completed, the installation and configuration of Apache and mod_dav on Fedora Core 4 is finished. The test outlined in Section F of this appendix should be performed to ensure that the installation was performed correctly.

C. XTS SINGLE LEVEL INSTALLATION

All of the procedures in this section are to be performed as the "admin" user on the XTS-400. These procedures download, install and configure the Apache 1.3.34 web server along with the mod_dav 1.0.3 Apache module. For this section, a base install of the MYSEA package on an XTS-400 running the STOP 6.1 operating system is assumed.

Step 1. Download the Apache server and the mod_dav module from the installation CD. A special tool, named *cdtool*, is used on the XTS-400 to transfer files from a CD. Unless otherwise noted, the steps in this section must be performed at the security level sl1:il3 since that is the configured level of the network interface used by the server.

(set security and integrity levels – min:oss)

```
cd /home/admin
cdtool cp /apache_1.3.34.tar.gz ./apache_1.3.34.tar.gz
cdtool cp /mod_dav-1.0.3-1.3.6.tar.gz ./mod_dav-1.0.3-
     1.3.6.tar.gz
```

Step 2. Change the security level of the downloaded files to *sl1:il3*:

SAK

Enter command?	`fsm`
Enter request?	`change`
Enter pathname?	`/home/http/htdocs/dav/public/sl1il3`
Modify access level?	`yes`
Enter new security level?	`sl1`
Enter new integrity level?	`il3`
Is the level correct?	`yes`
Enter new owner name?	`admin`
Enter new group name?	`stop`
Modify discretionary access?	`no`
Display the object?	`no`
Okay to change?	`yes`

Step 3. Unpack the Apache server and the mod_dav module:

(set security and integrity levels – sl1:il3)

```
gunzip apache_1.3.34.tar.gz
tar xvf apache_1.3.34.tar
gunzip mod_dav-1.0.3-1.3.6.tar.gz
tar xvf mod_dav-1.0.3-1.3.6.tar
```

Step 4. Build and install mod_dav into the Apache source directory. While this is done before Apache is built and installed, the Apache configuration script must be run first to ensure that certain files required for mod_dav configuration are present.

```
cd /home/admin/apache_1.3.34
./configure --prefix=/home/sldav/apache
cd /home/admin/mod_dav-1.0.3-1.3.6
./configure --with-apache=/home/admin/apache_1.3.34
make
make install
```

Step 5. Build and install Apache:

```
cd /home/admin/apache_1.3.34
./configure --prefix=/home/admin/apache --activate-
     module=src/modules/dav/libdav.a
make
make install
```

Step 6. Save and edit the default Apache configuration:

```
cd /home/sldav/apache/conf/
cp httpd.conf httpd.conf.bak
vi httpd.conf
```

Step 7. Append the following text to the end of the “httpd.conf” file:

```
DAVLockDB /home/admin/apache/var/DAVLock
<Location /dav>
     DAV On
</Location>
```

Step 8. Create the two directories referenced in the configuration lines above:

```
mkdir /home/admin/apache/var
mkdir /home/admin/apache/htdocs/dav
```

Step 9. Before performing this step, make sure that the network interfaces and daemons have been started using the trusted *startup* command.

```
/home/admin/apache/bin/apachectl start
```

Once all of these steps have been completed, the installation and configuration of Apache and mod_dav at a single level on the XTS-400 is finished. The test outlined in Section F of this appendix should be performed to ensure that the installation was performed correctly.

D. XTS MULTILEVEL INSTALLATION

All of the procedures in this section are to be performed as the "admin" user on the XTS-400, except where otherwise noted. These procedures download, install and configure the mod_dav 1.0.3 Apache module with the existing Apache 1.3.34 installation. For this section, a base install of the MYSEA package on an XTS-400 running the STOP 6.1 operating system is assumed.

Step 1. Download and unpack the Apache server and the mod_dav module from the installation CD. A special tool, named *cdtool*, is used on the XTS-400 to transfer files from a CD.

(set security and integrity levels – min:oss)

```
cd /home/admin
cdtool cp /dev/cdrom /mod_dav-1.0.3-1.3.6.tar.gz ./mod_dav-1.0.3-
     1.3.6.tar.gz
gunzip mod_dav-1.0.3-1.3.6.tar.gz
exit
```

(set security and integrity levels – sl0:il3)

```
cd /usr/local/mysea/apache
tar xvf /home/admin/mod_dav-1.0.3-1.3.6.tar
mv mod_dav-1.0.3-1.3.6 mod_dav
```

Step 2. Edit the Apache install configuration:

```
cd /usr/local/mysea/apache/src
vi Configuration
```

Step 3. Append the following text to the end of the “Configuration” file:

```
AddModule modules/dav/libdav.a
```

Step 4. Configure mod_dav. While this is done before Apache is built, the Apache configuration script needs to be run first to ensure that certain files required for mod_dav configuration are present.

```
./Configure
cd /usr/local/mysea/apache/mod_dav
./configure --with-apache=/usr/local/mysea/apache
```

Step 5. Edit the mod_dav makefile. This is necessary to tell mod_dav where to look for special MYSEA source code headers.

```
vi Makefile
```

Step 6. Find the line that starts with “ALL_CFLAGS = $(CPPFLAGS)” and add the following text immediately afterwards on the same line:

```
-I/usr/local/mysea/include
```

Step 7. Build and install mod_dav:

```
make
make install
```

Step 8. Build Apache:

```
cd /usr/local/mysea/apache/src
./Configure
make
```

Step 9. Save and edit the default Apache configuration:

```
cd /home/http/conf/
cp httpd.conf httpd.conf.bak
vi httpd.conf
```

Step 10. Append the following text to the end of the “httpd.conf” file:

```
DAVLockDB /tmp/DAVLock
<Location /dav>
      DAV On
</Location>
```

Step 11. Create the following default WebDAV directories for both the public and private data stores:

```
mkdir /home/http/htdocs/dav
mkdir /home/http/htdocs/dav/public
mkdir /home/http/htdocs/dav/public/sl0il3
mkdir /home/http/htdocs/dav/public/sl1il3
mkdir /home/http/htdocs/dav/public/sl2il3
mkdir /home/http/htdocs/dav/public/sl3il3
mkdir /home/http/htdocs/dav/public/sl4il3
mkdir /home/http/htdocs/dav/public/sl5il3
mkdir /home/http/htdocs/dav/public/sl6il3
mkdir /home/http/htdocs/dav/public/sl7il3
mkdir /home/http/htdocs/dav/private
```

Step 12. Set all of the directories created in the previous step to the proper security levels. This is done in the trusted environment with the trusted *fsm* command. Repeat this process to set each directory in “/home/http/htdocs/dav/public/” to the security level described by its name:

(set security and integrity levels – sl0:il3)
SAK

Enter command?	`fsm`
Enter request?	`change`
Enter pathname?	`/home/http/htdocs/dav/public/sl1il3`
Modify access level?	`yes`
Enter new security level?	`sl1`
Enter new integrity level?	`il3`
Is the level correct?	`yes`
Enter new owner name?	`admin`
Enter new group name?	`stop`
Modify discretionary access?	`no`
Display the object?	`no`
Okay to change?	`yes`

Step 13. For each user who needs to have access to a private area via WebDAV, perform the following steps. The username “demo” is used as an example here and can be replaced with any valid username.

```
ln -s /home/demo /home/http/htdocs/dav/private/demo
```

(log in as “demo” and set security and integrity levels – sl0:il3)

```
mkdir /home/demo/sl0il3
mkdir /home/demo/sl1il3
mkdir /home/demo/sl2il3
mkdir /home/demo/sl3il3
mkdir /home/demo/sl4il3
mkdir /home/demo/sl5il3
mkdir /home/demo/sl6il3
mkdir /home/demo/sl7il3
```

Step 14. Set all of the directories created in step 10 to the proper security levels. This is done in the trusted environment with the trusted *fsm* command. Again, the user “demo” is used as an example. Repeat the following process to set each directory in “/home/demo” to the security level described by its name:

(set security and integrity levels – sl0:il3)

SAK

Enter command?	`fsm`
Enter request?	`change`
Enter pathname?	`/home/demo/sl1il3`
Modify access level?	`yes`
Enter new security level?	`sl1`
Enter new integrity level?	`il3`
Is the level correct?	`yes`
Enter new owner name?	`demo`
Enter new group name?	`other`
Modify discretionary access?	`yes`
Enter object modes for owner?	`rwx`
Enter user name?	`<CR>`
Enter object modes for group?	`none`
Enter group name?	`<CR>`
Enter object modes for others?	`none`
Display the object?	`no`
Okay to change?	`yes`

Once all of these steps have been completed, the installation and configuration of mod_dav in the multilevel environment on the XTS-400 is finished. The test outlined in

Section F of this appendix should be performed to ensure that the installation was performed correctly.

E. CADAVER INSTALLATION

All of the procedures in this section are to be performed on a client machine running Fedora Core 4. These procedures download and install the Cadaver 0.22.2 WebDAV client from the installation CD.

```
cp /media/cdrom/cadaver-0.22.2.tar ./
tar -xvf cadaver-0.22.2.tar
cd cadaver-0.22.2
./configure
make
make install
```

F. INSTALLATION TESTING

The following procedures are designed to test whether or not the installation of mod_dav into Apache was successful. Each step is to be performed on the operating system specified.

Step 1. Create a file on the machine hosting the WebDAV server in the *root DAV directory* and add some arbitrary text:

```
vi readme.txt
```

Step 2. Access the file created in Step 1 with a WebDAV client on one or more operating systems. Section G of this appendix contains instructions for using different WebDAV clients to connect to a WebDAV server. If testing the MLS WebDAV server, log in and establish a session with either the TPE or TCBE client. Using one of the graphical WebDAV clients, simply double-click on “readme.txt” in the WebDAV directory to ensure that it is accessible. To perform the same check with Cadaver, enter the following command into the Cadaver command line:

```
edit readme.txt
```

Step 3. Attempt to upload a new file to the *root DAV directory*. Once again, connect to the WebDAV server using the instructions in Section G of this appendix. Using one of the graphical WebDAV clients, create a new text file named “upload.txt”

and add some arbitrary text to it. Next drag the new “upload.txt” file to the connected WebDAV directory. To perform the same check with Cadaver, enter the following command into the Cadaver command line and add some arbitrary text:

```
edit upload.txt
```

Step 4. Ensure that “upload.txt” exists on the machine hosting the WebDAV server and that it contains the proper text. The command to check the contents of the file is:

```
vi upload.txt
```

If all of the above procedures were successfully performed, then the installation of mod_dav has been successful.

G. WEBDAV CLIENT INSTRUCTIONS

This section provides instructions for using four different WebDAV clients to connect to a WebDAV server. For each of these sets of instructions, assume the IP address of the WebDAV server is 192.168.0.130, since this is the default address of the port interfacing with the MLS network on the MYSEA server.

1. Windows XP

From any Window Explorer window, type the following into the address bar:

```
\\192.168.0.130\dav
```

The connection is now established and the WebDAV directory can be accessed in the same manner as a normal directory.

2. Fedora Core 4 Linux

Select the “Connect to Server…” option from the “Places” menu and enter the following options:

Service: `WebDAV (HTTP)`
Server: `192.168.0.130`
Port: `80`
Folder: `dav`

The connection is now established and the WebDAV directory can be accessed in the same manner as a normal mounted network drive.

3. Macintosh OS 10.3

From the Finder, select the “Connect to Server…” option from the “Go” menu and type the following in the address bar:

```
http://192.168.0.130/dav
```

The connection is now established and the WebDAV directory can be accessed in the same manner as a normal mounted network drive.

4. Cadaver

From terminal, type the following command:

```
cadaver 192.168.0.130/dav
```

The connection is now established and the WebDAV directory can be accessed in a similar manner as a normal FTP session.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: TEST PROCEDURES

These procedures describe the steps to perform the tests outlined in Chapter IV. Each test falls into two categories: graphical clients and Cadaver. The graphical clients are those clients that are built in to Fedora Core 4, Windows XP and Mac OS X. These graphical clients are accessed through the graphical user interface (GUI) of their host operating systems. Cadaver is a command line WebDAV client that can be installed on any UNIX based operating system using the instructions in Appendix A. Instructions for using all of these clients are found in Appendix A.

A. FUNCTIONAL TEST PROCEDURES

These tests are to be performed on all three of the installation types described in Appendix A. All of these tests assume that the file "readme.txt" exists on the WebDAV server in the *root DAV directory*, and that "readme.txt' and "upload.txt" exists on the desktop of the client system used for testing. Each of these files should contain some uniquely identifying text to ensure that operations were indeed successful. Each of these tests should be able to be successfully performed, unless stated otherwise.

Test A1 intended to ensure that files can be read via the WebDAV client.

Graphical Clients:

Double click on "readme.txt" in the DAV directory.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
edit readme.txt
```

Test A2 intended to ensure that files can be written via the WebDAV client.

Graphical Clients:

Drag the file "readme.txt" from the desktop to the DAV directory.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
edit readme.txt
```

Add some arbitrary text to the file and save it.

Test A3 ensures that files can be created via the WebDAV client.

Graphical Clients:

Drag the file "upload.txt" from the desktop to the DAV directory.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
edit upload2.txt
```

Add some arbitrary text to the file and save it.

Test A4 ensures that files can be locked via the WebDAV client.

Graphical Clients:

Files cannot be locked using any of the graphical clients.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
lock upload2.txt
```

Test A5 ensures that files can be unlocked via the WebDAV client.

Graphical Clients:

Files cannot be unlocked using any of the graphical clients.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
unlock upload2.txt
```

Test A6 ensures that files can be deleted via the WebDAV client.

Graphical Clients:

Select the file "upload.txt" in the DAV directory and press the "delete" key.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
rm upload2.txt
```

Test A7 ensures that directories can be created via the WebDAV client.

Graphical Clients:

Create a new directory named "test" in the DAV directory.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
mkdir test2
```

Test A8 ensures that directories can be traversed via the WebDAV client.

Graphical Clients:

Double click on the directory "test" in the DAV directory.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
cd test2
```

Test A9 ensures that directories can be deleted via the WebDAV client.

Graphical Clients:

Select the directory "test" in the DAV directory and press the "delete" key.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
rmcol test2
```

Test A10 ensures that no unrecoverable errors occur when attempting to read a file that does not exist.

Graphical Clients:

This test is not applicable for a graphical client.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
cat ifile
```

Test A11 ensures that no unrecoverable errors occur when attempting to delete a file that does not exist.

Graphical Clients:

This test is not applicable for a graphical client.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
rm ifile
```

Test A12 ensures that no unrecoverable errors occur when attempting to lock a file that does not exist.

Graphical Clients:

This test is not applicable for a graphical client.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
lock ifile
```

Test A13 ensures that no unrecoverable errors occur when attempting to unlock a file that does not exist.

Graphical Clients:

This test is not applicable for a graphical client.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
unlock ifile
```

Test A14 ensures that no unrecoverable errors occur when attempting to delete a directory that does not exist.

Graphical Clients:

This test is not applicable for a graphical client.

Cadaver:

Issue the following command in the DAV directory with Cadaver:

```
rmcol idir
```

B. SINGLE LEVEL SECURITY TEST PROCEDURES

These tests are to be performed on the single level XTS installation type described in Appendix A. All of these tests assume that the file "readme.txt" exists on the WebDAV server in the *root DAV directory* as well as in all of the subdirectories (i.e. the directories named for their security levels), and that "readme.txt' and "upload.txt" exists on the desktop of the client system used for testing. Each of these files should contain some uniquely identifying text to ensure that operations were indeed successful. The expected results of each of these tests are given inline with the test procedure. All of these tests are performed at the level *sl1:il3*.

Test B1 ensures that a user is not able to read data from a secrecy level higher than the level of the currently established session. This operation should fail.

Graphical Clients:

Navigate to the directory named “sl2il3”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd sl2il3
```

Test B2 ensures that a user is able to read data from a secrecy level lower than the level of the currently established session. This operation should succeed.

Graphical Clients:

Navigate to the directory named “sl0il3”.

Open the file named “readme.txt”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd sl0il3
cat readme.txt
```

Test B3 ensures that a user is able to read data from a secrecy level equal to the level of the currently established session. This operation should succeed.

Graphical Clients:

Navigate to the directory named “sl1il3”.

Open the file named “readme.txt”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd sl1il3
cat readme.txt
```

Test B4 ensures that a user is not able to write data to a secrecy level higher than the level of the currently established session. This operation should fail.

Graphical Clients:

Drag the file “upload.txt” from the desktop to the directory “sl2il3”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd sl2il3
```

Test B5 ensures that a user is able to write data to a secrecy level lower than the level of the currently established session. This operation should succeed.

Graphical Clients:

Drag the file “upload.txt” from the desktop to the directory “sl0il3”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd sl0il3
```

Test B6 ensures that a user is not able to write data to a secrecy level equal to the level of the currently established session. This operation should fail.

Graphical Clients:

Drag the file “upload.txt” from the desktop to the directory “sl1il3”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd sl1il3
edit upload2.txt
```

Add some arbitrary text to the file and save it.

C. MULTILEVEL SECURITY TEST PROCEDURES

These tests are to be performed on all the multilevel XTS installation type described in Appendix A. All of these tests assume that the file "readme.txt" exists on the WebDAV server in the *root DAV directory* as well as in all of the subdirectories of the

“public” directory (i.e. the directories named for their security levels), that the files “adminread.txt” (with the owner set to the “admin” user and the permissions set so that only the “admin” user can read the file) and “adminwrite.txt” (with the owner set to the “admin” user and the permissions set so that only the “admin” user can write but all users can read the file) exist in the subdirectories of the “public” directory and that "readme.txt' and "upload.txt" exists on the desktop of the client system used for testing. Each of these files should contain some uniquely identifying text to ensure that operations were indeed successful. The expected results of each of these tests are given inline with the test procedure. All of these test procedures are written with the assumed session level of *sl1:il3* and should be modified appropriately for other session levels. The session must be established as the demo user.

Due to an unresolved error, the contents of the “public” directory cannot be listed. In order to navigate to the subdirectories of the “public” directory via one of the graphical clients, it is necessary to go straight to that directory by name, “dav/public/sl1il3” for example. Instructions for accessing DAV directories by name are found in Section G of Appendix A.

Tests C1 through C3 ensure that a user is not able to read data from a secrecy level higher than the level of the currently established session. This operation should fail as the higher level directory is inaccessible.

Graphical Clients:

Go to the directory named “dav/public/sl2il3”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd public/sl2il3
```

Tests C4 through C6 ensure that a user is able to read data from a secrecy level lower than the level of the currently established session. This operation should succeed.

Graphical Clients:

Go to the directory named “dav/public/sl0il3”.

Open the file “readme.txt”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd public/sl0il3
cat readme.txt
```

Tests C7 through C9 ensure that a user is able to read data from a secrecy level equal to the level of the currently established session. This operation should succeed.

Graphical Clients:

Go to the directory named “dav/public/sl1il3”.

Open the file “readme.txt”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd public/sl1il3
cat readme.txt
```

Tests C10 through C12 ensure that a user is not able to write data to a secrecy level higher than the level of the currently established session. This operation should fail as the higher level directory is inaccessible.

Graphical Clients:

Go to the directory named “dav/public/sl2il3”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd public/sl2il3
```

Tests C13 through C15 ensure that a user is not able to write data to a secrecy level lower than the level of the currently established session. This operation should fail.

Graphical Clients:

Go to the directory named “dav/public/sl0il3”.

Drag the file “upload.txt” from the desktop to the directory “sl0il3”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd public/sl0il3
edit upload2.txt
```

Add some arbitrary text to the file and save it.

Tests C16 through C18 ensure that a user is able to write data to a secrecy level equal to the level of the currently established session. This operation should succeed.

Graphical Clients:

Go to the directory named “dav/public/sl1il3”.

Drag the file “upload.txt” from the desktop to the directory “sl1il3”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd public/sl1il3
edit upload2.txt
```

Add some arbitrary text to the file and save it.

Test C19 ensures that a user is not able to read data from a file that is owned by another user, has permissions set to allow only the owner to read and has a secrecy level lower than the level of the currently established session. This operation should fail.

Graphical Clients:

Go to the directory named “dav/public/sl0il3”.

Open the file named “adminread.txt”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd public/sl0il3
cat adminread.txt
```

Test C20 ensures that a user is not able to read data from a file that is owned by another user, has permissions set to allow only the owner to read and has a secrecy level equal to the level of the currently established session. This operation should fail.

Graphical Clients:

Go to the directory named “dav/public/sl1il3”.

Open the file named “adminread.txt”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd public/sl1il3
cat adminread.txt
```

Test C21 ensures that a user is not able to write data to a file that is owned by another user, has permissions set to allow only the owner to write and has a secrecy level equal to the level of the currently established session. This operation should fail.

Graphical Clients:

Go to the directory named “dav/public/sl1il3”.

Edit the file named “adminwrite.txt”.

Cadaver:

Issue the following commands in the DAV directory with Cadaver:

```
cd public/sl1il3
edit adminwrite.txt
```

Add some arbitrary text to the file and save it.

Test C22 ensures that multiple users are not able to lock the same file simultaneously. This operation should fail.

Graphical Clients:

This test is not applicable for a graphical client.

Cadaver:

Issue the following command with the first client in the DAV directory with Cadaver:

```
cd public/sl1il3
lock readme.txt
```

Issue the following command with the second client in the DAV directory with Cadaver:

```
cd public/sl1il3
lock readme.txt
```

Test C23 ensures that a user is not able to write to a file that has been locked by another user. This operation should fail.

Graphical Clients:

This test is not applicable for a graphical client.

Cadaver:

Issue the following command with the first client in the DAV directory with Cadaver:

```
cd public/sl1il3
lock readme.txt
```

Issue the following command with the second client in the DAV directory with Cadaver:

```
cd public/sl1il3
edit readme.txt
```

LIST OF REFERENCES

1. Goland, Y., Whitehead, E., Faizi, A., Carter, S., Jensen, D., “HTTP Extensions for Distributed Authoring -- WEBDAV,” Request for Comments: 2518, February 1999. Available: http://www.ietf.org/rfc/rfc2518.txt, Accessed: January 2006

2. Jennings, C., Hardie, T., Hollenbeck, S., “WWW Distributed Authoring and Versioning (webdav),” IETF Charter, June 2005. Available: http://www.ietf.org/html.charters/webdav-charter.html Accessed: January 2006

3. Stein, G., “mod_dav: a DAV module for Apache,” WebDAV Official Website, November 2001. Available: http://webdav.org/mod_dav/ Accessed: February 2006

4. Irvine, C. E., Levin, T. E., Nguyen, T. D., Shifflett, D. J., Khosalim, J., Clark, P. C., Wong, A., Afinidad, F., Bibighaus, D., and Sears, J., "Overview of a High Assurance Architecture for Distributed Multilevel Security", Proceedings of the 2004 IEEE Systems, Man and Cybernetics Information Assurance Workshop, West Point, NY, June 2004.

5. Nguyen, T.D., Levin, T. E., Irvine, C. E., “MYSEA Testbed,” Proc. 6th IEEE Systems, Man and Cybernetics Information Assurance Workshop, West Point, NY, June 2005, pp. 438-439.

6. Egan, M., "An Implementation Of Remote Application Support In A Multilevel Environment", Master’s Thesis, Naval Postgraduate School, Monterey, CA, March 2006

7. Cadaver 0.22.3 – Available: http://webdav.org/cadaver/cadaver-0.22.3.tar.gz Accessed: April 2006

8. Apache 1.3.34 – Available: http://archive.apache.org/dist/httpd/apache_1.3.34.tar.gz Accessed: April 2006

9. mod_dav 1.0.3 – Available: http://webdav.org/mod_dav/mod_dav-1.0.3-1.3.6.tar.gz Accessed: April 2006

10. Fedora Core 4 – Available: http://www.redhat.com/fedora/ Accessed: April 2006

11. National Information Assurance Partnership (NIAP). “XTS-400™ / STOP™ 6.1.E,” March 2005. Available: http://niap.nist.gov/cc-scheme/st/ST_VID3012a.html. Accessed: 6/8/2006.

12. Apache 1.3.12 – Available: http://archive.apache.org/dist/httpd/apache_1.3.12.tar.gz Accessed: April 2006

13. McNamee, D., Heller, S., Huff, D., “Building Multilevel Secure Web Services-Based Components for the Global Information Grid”, STSC CrossTalk, May 2006.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Hugo A. Badillo
   NSA
   Fort Meade, MD

4. George Bieber
   OSD
   Washington, DC

5. RADM Joseph Burns
   Fort George Meade, MD

6. John Campbell
   National Security Agency
   Fort Meade, MD

7. Deborah Cooper
   DC Associates, LLC
   Roslyn, VA

8. CDR Daniel L. Currie
   PMW 161
   San Diego, CA

9. Louise Davidson
   National Geospatial Agency
   Bethesda, MD

10. Steve Davis
    NRO
    Chantilly, VA

11. Vincent J. DiMaria
    National Security Agency
    Fort Meade, MD

12. CDR James Downey
NAVSEA
Washington, DC

13. Dr. Diana Gant
National Science Foundation

14. Jennifer Guild
SPAWAR
Charleston, SC

15. Richard Hale
DISA
Falls Church, VA

16. CDR Scott D. Heller
SPAWAR
San Diego, CA

17. Wiley Jones
OSD
Washington, DC

18. Russell Jones
N641
Arlington, VA

19. David Ladd
Microsoft Corporation
Redmond, WA

20. Dr. Carl Landwehr
DTO
Fort George T. Meade, MD

21. Steve LaFountain
NSA
Fort Meade, MD

22. Dr. Greg Larson
IDA
Alexandria, VA

23. Dr. Karl Levitt
NSF
Arlington, VA

24. Dr. Vic Maconachy
NSA
Fort Meade, MD

25. Doug Maughan
Department of Homeland Security
Washington, DC

26. Dr. John Monastra
Aerospace Corporation
Chantilly, VA

27. John Mildner
SPAWAR
Charleston, SC

28. Mark T. Powell
Federal Aviation Administration
Washington, DC

29. Jim Roberts
Central Intelligence Agency
Reston, VA

30. Jon Rolf
NSA
Fort Meade, MD

31. Keith Schwalm
Good Harbor Consulting, LLC
Washington, DC

32. Charles Sherupski
Sherassoc
Round Hill, VA

33. Ken Shotting
NSA
Fort Meade, MD

34. CDR Wayne Slocum
SPAWAR
San Diego, CA

35. Dr. Ralph Wachter
ONR
Arlington, VA

36. David Wirth
N641
Arlington, VA

37. CAPT Robert Zellmann
CNO Staff N614
Arlington, VA

38. Dr. Cynthia E. Irvine
Naval Postgraduate School
Monterey, CA

39. Thuy D. Nguyen
Naval Postgraduate School
Monterey, CA

40. Jeremiah A. Bradney
Civilian, Naval Postgraduate School
Monterey, CA